In [ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```python
df = pd.read_csv('BBC News.csv')
df.head()
```

Out[ ]:

	ArticleId	Text	Category
0	1833	worldcom ex-boss launches defence lawyers defe...	business
1	154	german business confidence slides german busin...	business
2	1101	bbc poll indicates economic gloom citizens in ...	business
3	1976	lifestyle governs mobile choice faster bett...	tech
4	917	enron bosses in $168m payout eighteen former e...	business

In [ ]:
```python
df.shape
```

Out[ ]:
```
(1490, 3)
```

In [ ]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1490 entries, 0 to 1489
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ArticleId  1490 non-null   int64
 1   Text       1490 non-null   object
 2   Category   1490 non-null   object
dtypes: int64(1), object(2)
memory usage: 35.0+ KB
```

In [ ]:
```python
df['Category'].value_counts()
```

Out[ ]:
```
sport            346
business         336
politics         274
entertainment    273
tech             261
Name: Category, dtype: int64
```

In [ ]:
```python
from gensim.utils import simple_preprocess
```

In [ ]:
```python
df['Text'][0]
```

Out[ ]:
```
'worldcom ex-boss launches defence lawyers defending former worldcom chief bernie ebbers against a battery of
fraud charges have called a company whistleblower as their first witness.  cynthia cooper  worldcom s ex-head
of internal accounting  alerted directors to irregular accounting practices at the us telecoms giant in 2002.
her warnings led to the collapse of the firm following the discovery of an $11bn (£5.7bn) accounting fraud. mr
ebbers has pleaded not guilty to charges of fraud and conspiracy.  prosecution lawyers have argued that mr ebb
ers orchestrated a series of accounting tricks at worldcom  ordering employees to hide expenses and inflate re
venues to meet wall street earnings estimates. but ms cooper  who now runs her own consulting business  told a
jury in new york on wednesday that external auditors arthur andersen had approved worldcom s accounting in ear
ly 2001 and 2002. she said andersen had given a  green light  to the procedures and practices used by worldco
m. mr ebber s lawyers have said he was unaware of the fraud  arguing that auditors did not alert him to any pr
oblems.  ms cooper also said that during shareholder meetings mr ebbers often passed over technical questions
to the company s finance chief  giving only  brief  answers himself. the prosecution s star witness  former wo
rldcom financial chief scott sullivan  has said that mr ebbers ordered accounting adjustments at the firm  tel
ling him to  hit our books . however  ms cooper said mr sullivan had not mentioned  anything uncomfortable  ab
out worldcom s accounting during a 2001 audit committee meeting. mr ebbers could face a jail sentence of 85 ye
ars if convicted of all the charges he is facing. worldcom emerged from bankruptcy protection in 2004  and is
now known as mci. last week  mci agreed to a buyout by verizon communications in a deal valued at $6.75bn.'
```

In [ ]:
```python
print(simple_preprocess(df['Text'][0])[:30])
```

```
['worldcom', 'ex', 'boss', 'launches', 'defence', 'lawyers', 'defending', 'former', 'worldcom', 'chief', 'berni
e', 'ebbers', 'against', 'battery', 'of', 'fraud', 'charges', 'have', 'called', 'company', 'whistleblower', 'a
s', 'their', 'first', 'witness', 'cynthia', 'cooper', 'worldcom', 'ex', 'head']
```

In [ ]:
```python
preprocessed_text = df['Text'].apply(lambda x: simple_preprocess(x))
```

In [ ]:
```python
preprocessed_text.head()
```

```
Out[ ]: 0    [worldcom, ex, boss, launches, defence, lawyer...
        1    [german, business, confidence, slides, german,...
        2    [bbc, poll, indicates, economic, gloom, citize...
        3    [lifestyle, governs, mobile, choice, faster, b...
        4    [enron, bosses, in, payout, eighteen, former, ...
        Name: Text, dtype: object
```

```
In [ ]: from gensim.models import Word2Vec as wtv
```

```
In [ ]: cbow_w2v_model = wtv(preprocessed_text, vector_size=300, window=6, min_count=3, sg=0)
        skgram_w2v_model = wtv(preprocessed_text, vector_size=300, window=6, min_count=3, sg=1)
```

```
In [ ]: print("cbow vocalulary size:", len(cbow_w2v_model.wv.index_to_key))
        print("skipgram vocalulary size:", len(skgram_w2v_model.wv.index_to_key))

cbow vocalulary size: 11639
skipgram vocalulary size: 11639
```

```
In [ ]: list(cbow_w2v_model.wv.key_to_index.items())[0:30]
```

```
Out[ ]: [('the', 0),
         ('to', 1),
         ('of', 2),
         ('and', 3),
         ('in', 4),
         ('for', 5),
         ('is', 6),
         ('that', 7),
         ('it', 8),
         ('on', 9),
         ('said', 10),
         ('was', 11),
         ('he', 12),
         ('be', 13),
         ('with', 14),
         ('has', 15),
         ('as', 16),
         ('have', 17),
         ('at', 18),
         ('by', 19),
         ('will', 20),
         ('but', 21),
         ('are', 22),
         ('from', 23),
         ('not', 24),
         ('they', 25),
         ('mr', 26),
         ('his', 27),
         ('an', 28),
         ('we', 29)]
```

```
In [ ]: cbow_w2v_model.wv.get_vector("the")
```

Out[ ]:
```
array([-8.31481293e-02, -1.08959831e-01,  3.16624165e-01,  1.54434587e-03,
       -7.74879873e-01, -1.43002719e-01,  5.98507762e-01,  1.75747007e-01,
       -4.23971564e-01, -1.15297869e-01,  1.05543986e-01, -2.19802976e-01,
       -2.92491969e-02, -5.98001387e-03, -3.73841748e-02,  1.61287144e-01,
        5.89107722e-02,  1.36541739e-01, -4.71634179e-01, -8.92911926e-02,
        6.26658350e-02,  2.89783478e-01,  2.92805254e-01,  2.22449020e-01,
        4.36693698e-01,  1.33230031e-01,  4.77638513e-01,  1.31512150e-01,
       -8.85784179e-02, -9.25805718e-02, -2.09146053e-01, -1.05642661e-01,
        4.29679424e-01, -1.05261125e-01,  3.63938957e-01, -1.00621849e-01,
        3.33613455e-01, -2.66896278e-01,  9.22922418e-02, -3.41482162e-01,
       -6.19196534e-01, -2.74881989e-01, -1.35366306e-01,  3.06177046e-03,
       -8.59389082e-02, -2.42521033e-01, -3.16761136e-01,  4.80453432e-01,
        2.66053379e-01,  1.36053950e-01,  2.70124581e-02,  3.92060757e-01,
       -4.35375243e-01, -3.44128579e-01,  6.05432615e-02, -2.84392715e-01,
        1.85792267e-01, -5.19332647e-01,  3.08807909e-01,  1.89534560e-01,
       -2.88624018e-01,  1.74403995e-01,  1.02405466e-01, -3.10110271e-01,
       -3.12393904e-01,  2.77542830e-01, -2.83166438e-01,  2.99043693e-02,
       -2.26354942e-01,  1.69878185e-01,  1.30848303e-01,  5.02552569e-01,
        3.63195866e-01, -6.84082985e-01,  3.20804894e-01, -8.36619586e-02,
        2.11873159e-01,  3.86077344e-01, -3.63968521e-01, -3.61642838e-02,
       -1.66195214e-01, -7.49792308e-02,  4.45948720e-01,  8.32359135e-01,
        1.62061870e-01,  2.31141075e-01,  2.95542330e-01,  6.46338686e-02,
        7.21007705e-01,  4.69092727e-01,  1.50736287e-01, -1.57407373e-02,
        5.74926853e-01, -3.43603224e-01,  6.93019450e-01,  6.26463950e-01,
       -2.48098969e-02, -5.68706870e-01, -8.00586119e-02,  1.00243354e+00,
       -1.12987362e-01,  6.11214399e-01, -5.55437095e-02,  2.09132001e-01,
        2.95554310e-01, -5.24666190e-01,  1.10862695e-01,  4.04500932e-01,
       -5.36414921e-01, -9.60293561e-02, -5.01584947e-01, -5.12956560e-01,
        4.07363735e-02, -1.14561707e-01,  2.57811636e-01,  3.90501857e-01,
        3.17558229e-01, -2.90058907e-02,  3.57310027e-01, -1.05664301e+00,
        4.87224221e-01, -3.86164218e-01,  5.62528074e-02, -3.70105028e-01,
        3.60096574e-01,  3.25710058e-01, -3.16010922e-01, -4.30547982e-01,
       -4.02816106e-03,  5.27020395e-01,  3.69689047e-01,  4.92151290e-01,
        2.24886745e-01, -3.07615936e-01,  2.93788433e-01,  2.82600343e-01,
        3.53133589e-01, -5.27369678e-02, -7.51898140e-02, -9.13511589e-03,
        3.20502132e-01, -5.25312424e-01, -3.60635161e-01,  2.73353189e-01,
        2.57427335e-01, -3.22402194e-02, -8.44493732e-02, -5.75938635e-02,
        1.63192704e-01, -1.76935405e-01,  1.94517881e-01, -4.19369608e-01,
       -2.40906969e-01,  8.91872197e-02, -1.63262337e-02,  2.51311809e-01,
        1.09999791e-01,  5.67877173e-01, -1.94887623e-01,  6.88980103e-01,
        5.87814301e-02,  6.38989329e-01, -1.66837782e-01,  1.21635579e-01,
        6.29232451e-02,  2.09149748e-01,  2.63677537e-01, -2.62093604e-01,
       -1.32405341e-01,  7.71947563e-01, -1.54958919e-01,  1.21768549e-01,
        7.37413880e-04, -2.31858522e-01, -6.84313923e-02, -2.12884098e-01,
       -7.75392413e-01, -7.04260707e-01, -2.23119795e-01,  8.31628125e-03,
       -8.89287665e-02,  3.72082829e-01,  7.82575607e-02, -1.79948106e-01,
       -4.17003095e-01,  2.27104127e-01,  5.27481914e-01,  4.03226539e-02,
        3.50736409e-01, -3.39316338e-01,  5.71545005e-01,  1.28546938e-01,
        3.80185246e-02, -2.79305458e-01, -2.61937082e-01, -3.15333158e-01,
        4.28935528e-01, -3.73215199e-01, -5.95132522e-02, -1.93054199e-01,
       -1.37388945e-01, -2.45019153e-01, -5.92831075e-01, -4.71529327e-02,
        4.55055058e-01, -1.59641039e-02, -4.39353913e-01,  1.75721824e-01,
        1.03957705e-01,  6.52221292e-02, -3.64449084e-01, -3.15247655e-01,
       -1.63018733e-01, -1.91016614e-01, -1.09987296e-01, -1.77198455e-01,
        1.54237583e-01,  1.44812584e-01, -5.22272706e-01, -6.16483092e-01,
       -2.26114653e-02,  1.93921879e-01, -3.83501351e-02, -6.45834744e-01,
       -2.67480891e-02, -5.14259100e-01, -1.65488213e-01,  4.27637309e-01,
       -4.92451817e-01, -6.03203429e-03,  6.45852908e-02, -2.96439141e-01,
       -1.41725451e-01,  6.11466914e-02, -8.11973512e-01, -1.19317017e-01,
       -1.64498284e-01,  2.34566648e-02, -8.94316733e-02, -1.95570007e-01,
       -6.11311384e-02,  8.69261697e-02, -1.99339673e-01,  4.85149659e-02,
        4.70284075e-01,  9.17656794e-02, -9.05571058e-02, -2.39461377e-01,
        2.93704003e-01,  3.32365632e-01,  5.73104657e-02, -1.38362959e-01,
       -4.51785736e-02, -4.91279662e-01, -5.92094660e-01, -1.52097419e-01,
        1.36717796e-01,  2.97354199e-02, -5.67434251e-01, -2.41632715e-01,
       -8.05434734e-02,  5.02280109e-02,  3.33774894e-01, -3.38742971e-01,
       -7.22909510e-01,  4.02991772e-01,  6.27576232e-01, -1.60528734e-01,
       -7.92566240e-01,  2.30318099e-01, -3.67947042e-01,  4.63940442e-01,
        3.95653784e-01,  4.41798568e-01, -2.24084854e-01, -1.99771985e-01,
        6.97567523e-01,  8.16607401e-02, -6.33180261e-01,  1.48243755e-02,
        1.49172684e-02,  5.86358570e-02, -3.02452624e-01,  3.12219203e-01,
       -1.77827448e-01, -6.39973760e-01, -3.11792850e-01, -9.55743715e-02,
       -1.13881208e-01, -1.38867170e-01, -1.12106018e-01, -1.32125318e-01,
        1.00709200e-01, -4.91553485e-01, -3.60093981e-01, -1.35394827e-01,
        2.94675112e-01,  5.83081215e-04,  3.97491813e-01, -1.19094707e-01],
      dtype=float32)
```

In [ ]:
```
cbow_w2v_model.wv.most_similar('oil')
```

```
Out[ ]: [('stock', 0.9794525504112244),
 ('exchange', 0.9731940031051636),
 ('profits', 0.9716687202453613),
 ('reserves', 0.9711026549339294),
 ('shares', 0.9663559198379517),
 ('india', 0.9648077487945557),
 ('surged', 0.9644091725349426),
 ('annual', 0.964008629322052),
 ('revenues', 0.9632318019866943),
 ('china', 0.9612467288970947)]
```

```
In [ ]: skgram_w2v_model.wv.most_similar('oil')
```

```
Out[ ]: [('gas', 0.9029224514961243),
 ('fuel', 0.8806427717208862),
 ('currency', 0.8786510229110718),
 ('steel', 0.8776251077651978),
 ('rosneft', 0.8774250745773315),
 ('gm', 0.8729245066642761),
 ('soaring', 0.8722900152206421),
 ('telecoms', 0.8714026808738708),
 ('nestle', 0.8687661290168762),
 ('verizon', 0.8679018616676331)]
```

```
In [ ]: cbow_w2v_model.wv.most_similar('web')
```

```
Out[ ]: [('networks', 0.9882583618164062),
 ('online', 0.9871498346328735),
 ('computer', 0.9865508079528809),
 ('ways', 0.9860543012619019),
 ('pc', 0.9851844906806946),
 ('operators', 0.9851817488670349),
 ('camera', 0.9851146936416626),
 ('audio', 0.9848463535308838),
 ('data', 0.9842791557312012),
 ('cameras', 0.9838600754737854)]
```

```
In [ ]: skgram_w2v_model.wv.most_similar('web')
```

```
Out[ ]: [('uses', 0.9186863303184509),
 ('internet', 0.8994223475456238),
 ('search', 0.8986114263534546),
 ('via', 0.8979914784431458),
 ('surfers', 0.8974375128746033),
 ('addresses', 0.8967556357383728),
 ('logs', 0.8889086246490479),
 ('text', 0.8880167603492737),
 ('programs', 0.8870521783828735),
 ('engine', 0.8818630576133728)]
```

```
In [ ]: cbow_w2v_model.wv.most_similar('football')
```

```
Out[ ]: [('dream', 0.9780482053756714),
 ('secures', 0.9776517152786255),
 ('liverpool', 0.9771413803100586),
 ('draw', 0.9769200682640076),
 ('tremendous', 0.9767696857452393),
 ('finish', 0.9763832688331604),
 ('highbury', 0.9763008952140808),
 ('premiership', 0.9759109020233154),
 ('referee', 0.9756413698196411),
 ('tigers', 0.9755011200904846)]
```

```
In [ ]: model = cbow_w2v_model
```

```
In [ ]: def get_embeddimng_w2v(doc_tokens):
            embeddings = []
            for tok in doc_tokens:
                if tok in model.wv.index_to_key:
                    embeddings.append(model.wv.get_vector(tok))
            return np.mean(embeddings, axis=0)
```

```
In [ ]: X_x2v_model = preprocessed_text.apply(lambda x: get_embeddimng_w2v(x))
```

```
In [ ]: X_x2v_model
```

```
Out[ ]: 0       [0.011973189, 0.19467688, -0.04046378, 0.05234...
        1       [0.0018969477, 0.09894876, 0.0024450996, 0.042...
        2       [-0.05274923, 0.061471768, 0.028618021, 0.0735...
        3       [-0.11910302, 0.09020186, 0.040273443, 0.05346...
        4       [-0.01840922, 0.0839911, 0.025732774, 0.057198...
                                      ...
        1485    [-0.023653498, 0.1565971, -0.030683016, 0.0250...
        1486    [-0.08261965, 0.092795834, 0.03159496, 0.04627...
        1487    [-0.0064144568, 0.100432545, -0.007991844, 0.0...
        1488    [-0.07534904, 0.032702174, 0.06799838, 0.07823...
        1489    [-0.09131403, 0.09652467, 0.045668837, 0.05301...
        Name: Text, Length: 1490, dtype: object
```

```
In [ ]: X_df = pd.DataFrame(X_x2v_model.to_list())
```

```
In [ ]: X_df
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	...	290	
0	0.011973	0.194677	-0.040464	0.052348	-0.137587	-0.141337	0.258361	0.556020	0.007281	-0.094389	...	0.059960	0.273
1	0.001897	0.098949	0.002445	0.042633	-0.162184	-0.089396	0.246095	0.520570	-0.060274	-0.065952	...	0.089016	0.295
2	-0.052749	0.061472	0.028618	0.073584	-0.109754	-0.087750	0.205602	0.571432	-0.060820	-0.071915	...	0.092383	0.319
3	-0.119103	0.090202	0.040273	0.053464	-0.053153	-0.044308	0.157580	0.684285	-0.028988	-0.034751	...	0.169173	0.412
4	-0.018409	0.083991	0.025733	0.057199	-0.162890	-0.132351	0.260639	0.496307	-0.073116	-0.055969	...	0.067953	0.279
...	...	...	...	...	...	...	...	...	...	...	...	...	
1485	-0.023653	0.156597	-0.030683	0.025079	-0.131411	-0.108913	0.237678	0.564928	-0.068893	-0.113438	...	0.066225	0.292
1486	-0.082620	0.092796	0.031595	0.046275	-0.085910	-0.088700	0.199786	0.626359	-0.054800	-0.072426	...	0.115426	0.364
1487	-0.006414	0.100433	-0.007992	0.032878	-0.199827	-0.117580	0.313925	0.507905	-0.090025	-0.001751	...	0.130540	0.334
1488	-0.075349	0.032702	0.067998	0.078232	-0.095765	-0.094552	0.184554	0.584113	-0.049896	-0.029715	...	0.134542	0.363
1489	-0.091314	0.096525	0.045669	0.053011	-0.104612	-0.092792	0.201088	0.564678	-0.037859	-0.043760	...	0.118060	0.337

1490 rows × 300 columns

```
In [ ]: from sklearn.preprocessing import LabelEncoder
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score, f1_score
```

```
In [ ]: le = LabelEncoder()
        df['Category'] = le.fit_transform(df['Category'])
```

```
In [ ]: x_train, x_test, y_train, y_test = train_test_split(X_df, df['Category'], test_size=0.2, random_state=134)
```

```
In [ ]: from sklearn.naive_bayes import GaussianNB
```

```
In [ ]: gnb = GaussianNB()
        gnb.fit(x_train, y_train)
```

```
Out[ ]: ▾ GaussianNB
        GaussianNB()
```

```
In [ ]: y_pred_gnb = gnb.predict(x_test)
```

```
In [ ]: accuracy_score(y_test, y_pred_gnb)
```

```
Out[ ]: 0.7684563758389261
```

Google word2vec pretrained model

```
In [ ]:
```